NASA Technical Memorandum 82746

Application of Integration Algorithms in a Parallel Processing Environment for the Simulation of Jet Engines

Susan M. Krosel and Edward J. Milner
Lewis Research Center
Cleveland, Ohio

Prepared for the
Fifteenth Annual Simulation Symposium
Tampa, Florida, March 17-19, 1982

NASA

E-1059

APPLICATION OF INTEGRATION ALGORITHMS IN A PARALLEL PROCESSING ENVIRONMENT FOR THE SIMULATION OF JET ENGINES

Susan M. Krosel and Edward J. Milner
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio

Abstract. The development of digital dynamic simulations requires careful selection of an appropriate integration algorithm. This paper illustrates the application of predictor-corrector integration algorithms developed for the digital parallel processing environment. The algorithms are implemented and evaluated through the use of a software simulator which provides an approximate representation of the parallel processing hardware. Test cases which focus on the use of the algorithms are presented and a specific application using a linear model of a turbofan engine is considered. Results are presented showing the effects of integration step size and the number of processors on simulation accuracy. Real-time performance, inter-processor communication, and algorithm startup are also discussed.

INTRODUCTION

In recent years, there has been a growing need to obtain simulation models which represent physical systems over their entire operating range. An example of this is the jet engine. The demand for higher performance in these systems has resulted in increased system complexity and a need for more in-depth analysis of their dynamic behavior. There is an additional need for detailed system models to support the development of digital controls for these systems. In both the design and evaluation of these controls, simulations are frequently used.

Digital computers are used extensively for simulation because of their ease of programming, repeatibility of results, and to a large degree, the portability of the simulations. Digital simulations, such as GENENG, DYNGEN, and NNEP (References 1,2,3) provide the capability of predicting the steady-state and dynamic performance of a wide variety of gas turbine engine configurations. Digital computers, however, are limited in their usefulness for time-critical simulation applications by their inherent sequential execution of program instructions and serial computation within these instructions. In applications such as the validation of digital control hardware and software, the requirement for real-time

response of the simulation has necessitated either the use of large, dedicated computer systems with instruction cycle times in nanoseconds or the simplification of the model.

With the advent of and current advances in digital micro-computer technology, it is now possible to develop small, compact, computer systems for simulation. More importantly, it may now be possible to implement a detailed simulation model and still achieve real-time operation. This will permit the simulation to be used in a wide variety of applications including digital control system development, checkout, and troubleshooting as well as performance studies. One approach that has been proposed is to connect several microprocessors in a parallel arrangement and to provide a means of communication between the processors. The simulation is then partitioned over the several processors by dividing the system equations among the N processors forming the parallel digital system. However, partitioning necessitates a careful and thorough consideration of the dynamic coupling within the model to determine the optimal breakdown of the system functions. In some cases, inherent parallelism in the system may simplify the partitioning. The issue of how many processors to use then can be addressed. For efficient operation, the portions of the simulation that are allocated to the individual processors should use approximately the same amount of compute time per processor. This will insure correct updating of system variables and avoid wasted time in the calculation cycle. The updating of variables within the partitioned simulation will require not only careful timing considerations but also efficient data transfer between processors to avoid inadvertent phase shift.

The development of digital simulations, in general, depends on the selection and implementation of suitable numerical integration algorithms. These algorithms should provide for accurate and efficient solution of the differential equations that describe the system being simulated. For the gas turbine engine, these equations are typically nonlinear and involve multivariable functions that describe the performance of the engine's rotating components (fan, compressor, and turbines). In general, most of the computing time is used in the calculation of the system derivative function. Therefore, in the selection of an integration algorithm in a time-critical application, one must consider the number of derivative calculations associated with the algorithm. Much work has been done in the study of integration methods for a single processor system (References 4,5). The design and application of integration algorithms for a parallel-processing system depends on additional factors, such as: the number of processors, the method of partitioning the simulation, the inter-computer data transfer mechanism, the

computational speed of the processors, and the need to input and output simulation data (References 6,7). For example, it may be possible to partition a problem into linear and nonlinear parts and to use different integration algorithms on each part (Reference 8).

This paper discusses the application of parallel predictor-corrector algorithms (Reference 7) to the simulation of a typical turbofan engine. The simulation is intended to run on an MIMD (multiple instruction - multiple data) parallel processing system (Reference 9). A software simulator was used to provide an approximate representation of a parallel processing system and to evaluate the performance of the algorithms.

A first-order system and a second-order system were used to evaluate the algorithms. Each of these systems was excited by a unit step. The effects of the number of processors used and the integration stepsize on simulation accuracy, resolution, and stability were determined. Results are presented and discussed in the following sections.

AN ALTERNATIVE TO PARTITIONING

Miranker and Liniger (Reference 7) suggest a parallel predictor-corrector integration technique which merits consideration as an alternative to partitioning. No partitioning is required because the algorithms they present require that the entire simulation reside on each processor. Normally when using predictor-corrector integration, the current corrected value of a parameter is based on its current predicted value. The operations are sequentially regimented with the requirement that calculation of the current predicted value be completed before calculation of the current corrected value may begin. However the Miranker and Liniger algorithms predict and correct current values based on values already evaluated in a previous calculation cycle. Hence, prediction of some values and correction of others can take place simultaneously.

This charcteristic of the algorithm allows concurrent calculation to take place on parallel processors. Taking advantage of this assumed calculation power, the algorithms are able to operate on more than one integration time step during a single computer simulation update cycle. The update cycle time remains fixed; but since more than one integration time step, h, is calculated during this period, the simulation is effectively speeded up. Ideally, unless stability or accuracy problems arise, using a sufficiently large number of parallel processors should allow real-time operation. The relationships among the number of processors, system stability, and system accuracy are examined in detail in the RESULTS AND DISCUSSION section.

Consider a dynamic system described by equations of the form

$$y' = f(x,y), \qquad x > 0, \qquad y(0) = y_0 \tag{1}$$

Suppose that numerical solution is required at the mesh points $x_n = (n - 1)h$, $n = 1,2,\ldots$, where h is the step size. Miranker and Liniger propose solving this system by using predictor-corrector algorithms such as the following second-order, four-processor algorithm

$$y^P_{2n+2} = y^C_{2n-2} + 4hf^P_{2n} \tag{2}$$

$$y^P_{2n+1} = y^C_{2n-2} + \frac{3h}{2}\left(f^P_{2n} + f^P_{2n-1}\right) \tag{3}$$

$$y^C_{2n} = y^C_{2n-1} + \frac{h}{2}\left(3f^P_{2n} - 9f^P_{2n-1}\right) \tag{4}$$

$$y^C_{2n-1} = y^C_{2n-3} + 2hf^C_{2n-2} \tag{5}$$

where y_n is an approximation to y at x_n, y^P_n is the predicted value at x_n; y^C_n is the corrected value at x_n, $f^P_n = f\left(x_n, y^P_n\right)$ and $f^C_n = f\left(x_n, y^C_n\right)$. Predictor-corrector algorithms for one, two, and four processors were included in Reference 7. An eight-processor algorithm was derived by the authors based on the Miranker and Liniger methods. Second-order algorithms used for this study are listed in Table 1.

Figure 1 shows the allocation of the parallel predictor-corrector algorithm equations for the four-processor case. As shown in Figure 1, the algorithm for four processors calculates predicted values on two of the processors (A and B) and corrected values on the other two (C and D). This results in two outputs (i.e., corrected values) per calculation cycle. The outputs for the four processor algorithm are, in order, the corrected values from processor D and then from processor C. During this same calculation cycle, the predicted values are being calculated on processors B and A for one and two output times in the future respectively. These values then feed back into the algorithm for the updating of the corrected values in the next calculation cycle. In each new calculation cycle, the current values of the inputs are brought into the calculations on each processor. For each state of the system being simulated, eleven transfers of data must be accomplished for the algorithm for each step in the integration cycle. It should be noted that only N-1 of the derivative function calculations are actually used in the N-processor algorithm.

SIMULATION APPROACH

To test the predictor-corrector algorithms before implementing them on actual parallel processor hardware, a software simulator was developed. The software simulator is written in Fortran to run on an IBM TSS 370. The program is structured in a modular fashion through the use of subroutines (Figure 2). Subroutines are used to represent the N processing elements. Pseudo-parallelism is achieved through the use of argument lists for variable transfer with distinct variable names during a calculation cycle. The code for the calculation of the system derivatives is contained in one subroutine. This permits easy modification of the software simulator for different systems being simulated. The software simulator accurately represents the problem-solving phase of the parallel processing predictor-corrector integration algorithms. Careful attention has been given to the outputting of results with respect to time. Representation of actual simulator control was not incorporated into the software simulator. Coding to represent a simulator controller and an input-output processor has been included at a very simplistic level.

RESULTS AND DISCUSSION

The Miranker and Liniger integration technique was examined from the real-time simulation point of view by applying it to a linear first-order system, two different linear second-order systems, and a linear fourth-order engine model. Following is a detailed examination of each.

FIRST-ORDER SYSTEM

The response of a first-order lag to a unit step input obeys the relationship

$$\tau y' + y = 1 \tag{6}$$

The closed-form solution to Equation (6) with the initial condition $y(0) = 0$ is

$$y = 1 - \exp(-x/\tau) \tag{7}$$

This response which is dependent on the value of the system time constant, τ, can be displayed as a single curve provided that the parameter x/τ is considered the independent variable. Rearranging Equation (6) yields

$$y' = \frac{1}{\tau}(1 - y) \tag{8}$$

Thus,

$$y' = f(\tau, y) = \frac{1}{\tau} g(y) \tag{9}$$

Hence, the second-order four processor predictor-corrector algorithm can be written as

$$y^P_{2n+2} = y^C_{2n-2} + 4\,\frac{h}{\tau}\,\dot{y}^P_{2n} \tag{10}$$

$$y^P_{2n+1} = y^C_{2n-2} + \frac{3}{2}\,\frac{h}{\tau}\left(\dot{y}^P_{2n} + \dot{y}^P_{2n-1}\right) \tag{11}$$

$$y^C_{2n} = y^C_{2n-3} - \frac{1}{2}\,\frac{h}{\tau}\left(3\dot{y}^P_{2n} - \dot{y}^P_{2n-1}\right) \tag{12}$$

$$y^C_{2n-1} = y^C_{2n-3} + 2\,\frac{h}{\tau}\,\dot{y}^C_{2n-2} \tag{13}$$

It is clear that the predictor-corrector solution (sequence of points) is dependent on the number of processors, the system time constant, τ, and the integration step size, h. For a fixed number of processors, the result is a family of solutions each corresponding to a different value of the parameter, h/τ.

To determine how well the predictor-corrector solutions compare with the closed-form solution (Equation (7)), a figure of merit was established in the following manner. The absolute value of the difference between the predictor-corrector solution and the closed-form solution was integrated over the time interval 0 to 4τ. This cumulative error was then expressed as a percentage of the total area bounded by the closed-form solution over that same time interval. This can be expressed by

$$\frac{\int_0^{4\tau} \left|\text{closed form - algorithm}\right| dx}{\int_0^{4\tau} \left|\text{closed form}\right| dx} \tag{14}$$

Thus, each of the predictor-corrector solutions has associated with it a relative error.

In studying the effects of varying the number of processors and/or the integration stepsize, it is helpful to define the "effective step advance per calculation cycle," H, as follows

$$H = h*(N/2) \tag{15}$$

where N is the number of parallel processors used. If only one processor is used, two derivative calculations are required per integration time step. One derivative calculation is required for the predictor and another one is required for the corrector. Consequently, the effective computation time for this case is twice the calculation time and the effective step advance per calculation cycle is h/2. In the four-processor case, only one derivative calculation per processor is required and each calculation cycle will result in two time steps being

calculated. Thus the effective step advance per calculation cycle is 2h. In studying the effect of changing the number of processors or using different integration step sizes, comparisons should be made on the basis of the same effective step advance per calculation cycle (that is, on the basis of equal values of H).

It was previously noted that the parameter h/τ could be used to eliminate the effect of the system time constant, τ, on the predictor-corrector solutions for a fixed number of processors. Similarly, the use of H/τ allows results from varying numbers of processors to be compared. The percent error resulting from using the second-order predictor-corrector algorithms as a function of the parameter H/τ is shown in Figure 3. Data corresponding to one, two, four, and eight processors are presented. We see that, for small values of H/τ, there is little, if any, advantage to using multiprocessors. A single processor gives good accuracy and does not require the transfer of any data. Hence, the mechanics of the simulation is kept simple. As the value of H/τ increases, we see that using a larger number of processors gives better accuracy. For a value of $H/\tau = 0.125$, accuracy is improved by almost a factor of four by using eight processors. This accuracy improvement is due in part to the larger number of points obtained from using more processors (a finer grid). When using eight processors, four points are calculated per calculation cycle. The effective step advance using eight processors consists of three intermediate points plus an end point. When using only two processors, one point is calculated per calculation cycle. In essence, then, when using two processors, all we are calculating in our solution are the end points of the eight processor case; and, hence, we are suffering a loss in accuracy by using a smaller number of processors.

As H/τ approaches a value of 0.25, we see that slightly improved accuracy is obtained using four processors rather than eight. This may be due to the step size becoming too large for the more complicated eight processor algorithm. For H/τ equal to 0.40, the eight processor algorithm is unstable. The integration step size has become too large for this complicated algorithm to hold together. However, the four processor algorithm is still stable at this point. This brings out an important point; namely, a simulation is not necessarily improved by using more processors.

For the purpose of illustration, suppose that a real-time simulation of a first-order system is desired. Suppose further that the corner frequency of the system is 50 radians/sec, that four processors are available for the simulation,

and that each processor takes 2 milliseconds to compute the required derivative function.

Hence, calculation time, t_c, is 0.002 sec and the time constant, τ = (corner frequency)$^{-1}$, is 0.02 sec. Since real-time simulation is desired, the effective step advance must be equal to the calculation time, t_c. Thus $H = t_c$ or $H/\tau = t_c/\tau = 0.002/0.02 = 0.10$. From Figure 3 we see that the corresponding simulation error will be approximately 1.5 percent if all four processors are used. Let resolution be defined as the calculated number of points per cycle at the highest frequency of interest. Then, resolution for this case is approximately 125 points/cycle at 50 radians/sec. Also from Figure 3 we see that the simulation error will be approximately 3 percent if one processor is used. Resolution for this case is still sufficient with approximately 30 points/cycle at 50 radians/sec. Now suppose that calculation time t_c increases from 2 milliseconds to 3.6 milliseconds. If 2 percent is the maximum allowable error and if four processors are available, we see from Figure 3 that a maximum value of $H/\tau = 0.13$ is permitted. However, for $t_c = 0.0036$ sec and $\tau = 0.02$ sec, $H/\tau = 0.18$ for real-time operation. Hence, the simulation cannot run real-time under these conditions.

What can be done to correct the situation? There are several alternatives. If eight processors were available, the simulation would run in real-time with 2.1 percent error with $H/\tau = 0.18$. If more than four processors were not available, then faster processors would be needed. If the hardware was fixed, some of the higher frequency dynamics of the simulation would have to be sacrificed for the sake of real-time operation. A time constant of $\tau = 0.028$ sec, which corresponds to a corner frequency of 36 radians/sec, would be required for real-time simulation with 2 percent error using four processors (for then $H/\tau = 0.0036/0.28 = 0.13$). In determining the number of processors to use to simulate a system with first-order characteristics, then, the choice depends on several factors. As we have seen, going to more processors gives better resolution and can give better accuracy. However, going to more processors may destabilize an otherwise stable simulation.

SECOND-ORDER SYSTEM

As a further test of the predictor-corrector algorithms, a second-order system consisting of a first-order lag feeding another first-order lag was simulated. The transfer function for such a system is $1/((\tau_1 s + 1)(\tau_2 s + 1))$ and its associated differential equation has the form

$$\tau_1\tau_2 y'' + (\tau_1 + \tau_2)y' + y = 1 \qquad (16)$$

For the initial conditions $y(0) = 0$ and $y'(0) = 0$, Equation (16) has the closed-form solution

$$y = 1 - \frac{\tau_1 e^{-x/\tau_1} - \tau_2 e^{-x/\tau_2}}{\tau_1 - \tau_2} \qquad (17)$$

For the study, two different cases were considered: CASE I has the values $\tau_1 = 0.01$ sec and $\tau_2 = 0.02$ sec; and CASE II has the values $\tau_1 = 0.002$ sec and $\tau_2 = 0.02$ sec; the procedure followed was similar to that used for the first-order system to obtain data. The second-order system responses can be reduced to a family of curves by using x/τ_1 as the independent variable. In this case, however, each curve represents a particular value of the ratio τ_2/τ_1. For the two cases, ($\tau_2/\tau_1 = 2$ and $\tau_2/\tau_1 = 10$) solutions for varying integration step size, h, were obtained using the predictor-corrector algorithms for one, two, four, and eight processors. As was done for the first-order system, an error measure was calculated for each solution. The time interval considered for this computation was 0 to four times the smaller time constant, τ_1. The error data for these cases were presented as a function of H/τ_1 to facilitate comparisons of results obtained from different numbers of processors.

Figure 4 presents the percent error versus H/τ_1 for CASE I. This system has the slower dynamics of the two cases considered ($\omega_1 = 100$ radians/sec and $\omega_2 = 50$ radians/sec). Notice that the error obtained using either one processor or two is virtually identical. However, using two processors gives better resolution (at least 15 points per cycle at the highest frequency compared with 8 points per cycle using one processor). Notice also that, for a value of $H/\tau_1 = 0.5$, one processor is stable and two or more are not.

If four processors can be used, improved accuracy and improved resolution are realized. At least 30 points per cycle are obtained at the highest frequency. Notice also that, at $H/\tau_1 = 0.4$, accuracy is improved 100 percent. Figure 4 shows that there is no reason to use eight processors. There is only a narrow range of stability for eight processors and both accuracy and resolution over this range are excellent with four processors.

Percent error versus H/τ_1 for CASE II is presented in Figure 5. This system has faster dynamics ($\omega_1 = 500$ radians/sec and $\omega_2 = 50$ radians/sec) than CASE I. CASE II also has a much smaller range of H over which the simulation is stable due to the higher frequencies involved. (The time constant τ_1 is five times smaller than in CASE I.) However, accuracy is excellent over the stable range, no matter how many processors are used. As in CASE I, the error

obtained using either one processor or two is virtually identical; however better resolution is obtained with two processors due to the finer spacing of points in the solution sequence. Resolution at 500 radians/sec, using one processor, is approximately 6 points/cycle. using two processors, it is approximately 12 points/cycle. Because of the excellent accuracy and resolution over the stable range for this case, there should be no need to use more than two processors.

Attaining real-time operation for a stiff system such as this would be difficult. The ω_1 high frequency is a dominant term requiring a very small integration time step for stability of the simulation. Since for real-time simulation $t_c = H$, we see that this requires the calculation cycle time to be no more than a millisecond. This calculation time may be met for elementary systems with available computers. However, for simulations of complex systems such as turbofan engines, to achieve real-time operation probably will require sacrificing some of the higher frequency dynamics. The most favorable condition for real-time simulation in the case is with one or two processors.

ENGINE MODEL

As a final test to determine whether the parallel predictor-corrector integration algorithms are applicable to the turbofan engine simulation problem, a state-space model of a representative engine was used as the system in the software simulator.

The model selected was a reduced-order (fourth-order) linear model at the sea-level, static, intermediate power operating condition. It was obtained from a full state (16th-order) linear model by normal reduction techniques (Reference 10). This linear model was validated along with other linear models of the engine in Reference 10. The reduced-order model still retains important dynamic characteristics of the system but is easier to handle mathematically. The mathematical representation of the system is given by the following equations

$$\dot{\bar{y}} = \bar{A}\,\bar{x} + \bar{B}\,\bar{u} \tag{18}$$

$$\bar{y} = \bar{C}\,\bar{x} + \bar{D}\,\bar{u} \tag{19}$$

where Equation (18) is a linear, constant-coefficient-matrix differential equation (valid only at a given operating condition) that represents the computation of state variable derivatives. Matrices $\bar{A}$, the system matrix, and $\bar{B}$, the control matrix, show the sensitivity of the time derivatives of the state variables $\dot{\bar{x}}$ to variations in the state variables $\bar{x}$ and control inputs $\bar{u}$. Equation (19) is a linear, constant-coefficient-matrix algebraic equation that represents the

computation of observed engine parameters. Matrices $\bar{C}$, the output matrix, and $\bar{D}$, the direct-coupie matrix, relate the changes in the observed parameters $\bar{y}$ to the variations in the state variables $\bar{x}$ and the control inputs $\bar{u}$. For the selected reduced-order model, the states, represented by vector $\bar{x}$, are fan speed, compressor speed, compressor discharge pressure, and interturbine pressure. The observed parameters, represented by vector $\bar{y}$, are engine net thrust, total engine airflow, burner-exit temperature, fan stall margin, compressor stall margin, measured fan-exit $\Delta p/p$ parameter, and calculated fan-exit $\Delta p/p$ parameter. The control inputs, represented by vector $\bar{u}$, are main burner fuel flow, exhaust nozzle jet area, fan inlet guide vane position, compressor variable vane position, and compressor bleed flow fraction.

The matrices $\bar{A}$, $\bar{B}$, $\bar{C}$, and $\bar{D}$ are given in Table II. No single linear model can accurately represent an engine over its entire operating range. Therefore, many linear models are typically derived at various flight conditions and power settings throughout the engine operating envelope. These models would be connected together in some manner to form a more accurate and representative simulation of the engine process. The selected engine model was derived at the sea-level, static, intermediate power operating condition and hence is valid only in that region.

To determine if the parallel predictor-corrector algorithms are suitable and appropriate integration methods for the engine simulation problem, a 3-percent step in fuel flow was input to the reduced-order model for one, two, four, and eight processor predictor-corrector algorithms. The resulting transients were compared with an exact solution obtained through evaluation of the state transition matrix.

In running the transient on the simulated multiprocessor systems, it was found that, as the number of processors increased, the integration step size for the algorithm had to be decreased. This decrease reflects the loss in stability due to these algorithms as more processors are used. Table III gives the tabulation of the number of processors and the maximum allowable timestep for stability. These results are summarized in Figure 6 which shows the effect of the number of processors on the required step size for stable operation. Data are presented for the first-order system, the second-order systems, and the engine model. In the engine model, Table II shows that the system possesses widely spaced eigenvalues, requiring the use of small timesteps for stability at the high frequency dynamics. However, Table III shows that the algorithms themselves require the use of smaller timesteps to assure the stability of the algorithm If real-time operation is desired, the decrease in the integration timestep means

that the time allowable to compute that system derivative also decreases. Table III also gives the time allowed for the derivative calculation for real-time operation.

The transient response of two states - fan speed and compressor discharge pressure - and two observed parameters - burner exit temperature and compressor stall margin - are shown in Figure 7 for the one, two, four, and eight processor cases and the exact solution. The transient is shown only for the first second and it is seen that agreement is quite good. After the first half second, the difference between the exact and any of the cases becomes insignificant. The only noticeable error is seen in the first tenth second of the transient. This is shown more clearly in Figure 8. This error at the beginning of the transient is caused by the startup delay inherent in the predictor-corrector algorithm. In general predictor-corrector integration algorithms are quite accurate provided a stable timestep is chosen. This has been shown in Figure 7. However, the distinct disadvantage that these algorithms possess is in their non-self-starting feature. This was shown in Figure 8 by emphasizing the first 0.1 second of the transient response. This is counterbalanced by the need for fewer evaluations of the system derivative, thereby requiring less computation time for the algorithm overall. If non-real-time operation is acceptable, then these parallel predictor-corrector algorithms may be useful.

Figure 9 gives a series of eight processor transients for varying timesteps of 0.0005, 0.00025, 0.0002, and 0.0001 second over a 0.5 second range. It can be seen that an integration step size of 0.0005 second produces oscillations in the first 0.3 second of the transient which subsequently die out. This is due to the instability of the algorithm at that step. The eight processor algorithm is stable when h is reduced to 0.00025 second. This implies that for real-time operation with this model, the evaluation of the derivative function must be completed in 1 millisecond (see Table III). Use of a smaller timestep does reduce these oscillations but results in timesteps much too small to have practical application.

Figure 10 shows a comparison of the predictor-corrector and actual responses over the first tenth of a second.

CONCLUSIONS

The Miranker and Liniger predictor-corrector algorithms can eliminate many problems associated with digital multiprocessing. The most obvious advantage is that the predictor-corrector algorithms do not require partitioning of the simulation model. In applying the algorithms to a linear first-order system and to

two different linear second-order systems, the errors obtained using one processor and two processors were virtually identical. In addition, using four processors generally cut this error in half. The range of stable timesteps was decreased by using more processors. In fact, in some cases, increasing the number of processors had a destabilizing effect. Using more processors did give better resolution but, in many cases, at the expense of decreasing the stability of the simulation.

Use of parallel predictor-corrector integration algorithms poses some disadvantage when real-time operation is desired. It has been shown that the use of these algorithms with increasing numbers of processors does result in a reduction in the stability of the algorithm and requires the use of smaller and smaller integration timesteps. This requirement of very small timesteps means that the computing device may have to perform more calculations over a specified time interval and be able to calculate the system derivative function in a shorter amount of time.

Also it has been seen that the predictor-corrector algorithms in themselves are not self-starting, therefore, either a deadtime occurs at the beginning of the transient (no response from the system) or some means of starting the algorithm must be devised. Since the calculation of the derivative is only required on N-1 processors (for the N-processor algorithm), this 'free calculation time' is available for implementation of a starting method. Proper switching logic must be incorporated to insure that the predictor-corrector algorithm takes over at the end of the startup period.

The parallel predictor-corrector integration algorithms may be a possible means to attain real-time operation for dynamic system simulation if:

(1) Microprocessor internal cycle times continue to decrease which will allow for faster calculation of the derivative functions;

(2) The cost of microprocessor memory continues to decrease allowing for the acquisition of enough memory for algorithm implementation;

(3) The system to be simulated does not have widely spaced time constants or very high dynamic frequencies such that the stable range of timesteps is not critically limited;

(4) Simplification of the simulation model, when necessary, can be made without excessive loss of accuracy.

In situations where parallel predictor-corrector integration algorithms cannot be used, other approaches such as problem partitioning and implicit integration should be considered.

REFERENCES

1. Koenig, R. W. and Fishbach, L. H., "GENENG - Program for Calculating Design Performance for Turbojet and Turbofan Engines," NASA TN D-6552, February 1972.

2. Sellers, J. F. and Daniele, C. J., "DYNGEN - A Program for Calculating Steady-State and Transient Performance of Turbojet and Turbofan Engines," NASA TN D-7901, April 1975.

3. Fishbach, L. H. and Caddy, M. J., "NNEP - The Navy NASA Engine Program," NASA TM X-71857, December 1975.

4. Benyon, P., "A Review of Numerical Methods for Digital Simulation," Simulation, November 1968.

5. Caranhan, B., Luther, H. A., and Wilkes, J. O., Applied Numerical Methods, John Wiley & Sons, Inc., 1969.

6. Halin, H. J., et al., "The ETH Multiprocessor Project: Parallel Simulation Continuous Systems," Simulation, October 1980.

7. Miranker, W. L. and Liniger, W., "Parallel Methods for the Numerical Integration of Ordinary Differential Equations," Mathematics of Computation, Vol. 21, 1967.

8. Palusinski, O. A. and Wait, J. V., "Simulation Methods of Combined Linear and Nonlinear Systems," Simulation, March 1978.

9. Flech, R. A. and Arpasi, D. J., "An Approach to Real-Time Simulation Using Parallel Processing," NASA TM-81731, 1981.

10. Daniele, C. J. and Krosel, S. M., "Generation of Linear Dynamic Models from a Digital Nonlinear Simulation," NASA TP-1388, February 1979.

TABLE I. - PARALLEL PREDICTOR-CORRECTOR INTEGRATION ALGORITHMS (ORDER TWO)

One processor:

$$y^P_{n+1} = y^C_n + \frac{h}{2}\left(3f^C_n - f^C_{n-1}\right)$$

$$y^C_{n+1} = y^C_n + \frac{h}{2}\left(f^P_{n+1} + f^C_n\right)$$

Two processor:

$$y^P_{n+1} = y^C_{n-1} + 2hf^P_n$$

$$y^C_n = y^C_{n-1} + \frac{h}{2}\left(f^P_n + f^C_{n-1}\right)$$

Four processor:

$$y^P_{2n+2} = y^C_{2n\ 2} + 4hf^P_{2n}$$

$$y^P_{2n+1} = y^C_{2n-2} + \frac{3h}{2}\left(f^P_{2n} + f^P_{2n-1}\right)$$

$$y^C_{2n} = y^C_{2n-3} - \frac{h}{2}\left(3f^P_{2n} - 9f^P_{2n-1}\right)$$

$$y^C_{2n-1} = y^C_{2n-3} + 2hf^C_{2n-2}$$

Eight processor:

$$y^P_{4n+4} = y^C_{4n-4} + 8hf^P_{4n}$$

$$y^P_{4n+3} = y^C_{4n-4} + \frac{7h}{2}\left(f^P_{4n} + f^P_{4n-1}\right)$$

$$y^P_{4n+2} = y^C_{4n-4} + 6hf^P_{4n-1}$$

$$y^P_{4n+1} = y^C_{4n-4} - \frac{5h}{2}\left(f^P_{4n} - 3f^P_{4n-1}\right)$$

$$y^C_{4n} = y^C_{4n-5} - \frac{5h}{2}\left(3f^P_{4n} - 5f^P_{4n-1}\right)$$

$$y^C_{4n-1} = y^C_{4n-6} - \frac{5h}{2}\left(3f^P_{4n-1} - 5f^P_{4n-2}\right)$$

$$y^C_{4n-2} = y^C_{4n-7} - \frac{5h}{2}\left(3f^P_{4n-2} - 5f^P_{4n-3}\right)$$

$$y^C_{4n-3} = y^C_{4n-7} - 4n\left(f^P_{4\ -3} - 2f^C_{4n-4}\right)$$

TABLE II. - ENGINE MODEL (4^{th} ORDER, SEA-LEVEL-STATIC, INTERMEDIATE)

Model equations:

$$\dot{\bar{x}} = \bar{A}\,\bar{x} + \bar{B}\,\bar{u}$$

$$\bar{y} = \bar{C}\,\bar{x} + \bar{D}\,\bar{u}$$

Model matrices:

System matrix, $\bar{A}$

	1	2	3	4
1	-3.684	-0.6320	103.0	-789.0
2	1.200	-6.137	79.95	-522.8
3	-1.659	5.819	-154.8	729.0
4	0.2584	-0.2028	5.581	-103.7

Control matrix, $\bar{B}$

	1	2	3	4	5
1	0.7260	-257.7	-107.9	-1.972	-0.1500D 05
2	0.6320	-458.4	13.40	-73.93	-0.1327D 05
3	0.8555	647.3	-16.04	59.29	0.7941D 05
4	0.1891D-01	-1229	10.16	-1.938	-399.6

Output matrix, C

	1	2	3	4
1	1.109	-0.8876	24.65	70.15
2	0.1382D-01	0.3152D-05	-0.7751D-07	-0.9650D-02
3	0.1093	-0.1181	1.136	-24.47
4	0.7315D-04	0.5169D-05	0.5422D-06	-0.1508D-01
5	-0.3066D-04	0.1294D-03	-0.2594D-02	0.1449D-01
6	0.4003D-04	-0.3016D-04	0.1166D-03	-0.1430D-01
7	0.1332D-04	0.4075D-05	-0.9470D-05	-0.3745D-02

Direct-couple matrix, D

	1	2	3	4	5
1	0.1647	-232.1	41.22	-8.291	-3459.
2	0.6344D-04	0.1782	0.7403	0.2073D-03	0.7989D-02
3	0.1329	-23.97	1.113	-1.199	-2464.
4	0.3239D-05	-0.3606D-02	-0.3220D-02	0.6736D-04	0.1239D-01
5	-0.8887D-08	0.1247D-01	-0.2915D-03	0.2437D-02	-0.1710D-01
6	0.3111D-07	-0.1163D-01	0.1737D-02	-0.3235D-03	0.5144D-02
7	0.4271D-07	-0.2992D-02	0.7025D-03	0.5175D-04	0.2557D-02

where:

The states represented by vector $\bar{x}$, are
- x1 Fan speed
- x2 Compressor speed
- x3 Compressor discharge pressure
- x4 Interturbine pressure

The output variables, represented by vector $\bar{y}$, are
- y1 Engine net thrust
- y2 Total engine airflow
- y3 Burner-exit temperature
- y4 Fan stall margin
- y5 Compressor stall margin
- y6 Empirical fan-exit $\Delta p/p$ parameter
- y7 Theoretical fan-exit $\Delta p/p$ parameter

The control inputs, represented by vector $\bar{u}$, are
- u1 Main-burner fuel flow
- u2 Exhaust-nozzle-jet area
- u3 Fan-inlet-guide-vane position
- u4 Compressor variable-vane position
- u5 Compressor bleed flow fraction

Model eigenvalues (sec^{-1})		Corresponding time constants (sec)
Real	Imaginary	
-3.1516863	0.000000	0.3173
-5.5467604	0.000000	0.1803
-60.662508	0.00000	0.0165
-198.91636	0.00000	0.0050

TABLE III. - CORRELATION AMONG NUMBER OF PROCESSORS, MAXIMUM INTEGRATION STEPSIZE FOR STABILITY, SYSTEM DERIVATIVE CALCULATIONS, OUTPUTS PER CYCLE, AND ALLOWABLE TIME FOR DERIVATIVE CALCULATION FOR REAL-TIME

Number of processors N	Maximum integration stepsize for stability h_{max} sec	Number of system derivative calculations per processor D	Number of outputs (corrector values) per cycle O	Allowable time for system derivative calculation for real-time operation given by $N/2 + h_{max}$ T_a sec
1	0.005	2	1	0.0025
2	.0025	1	1	.0025
4	.001	1	2	.002
8	.00025	1	4	.001

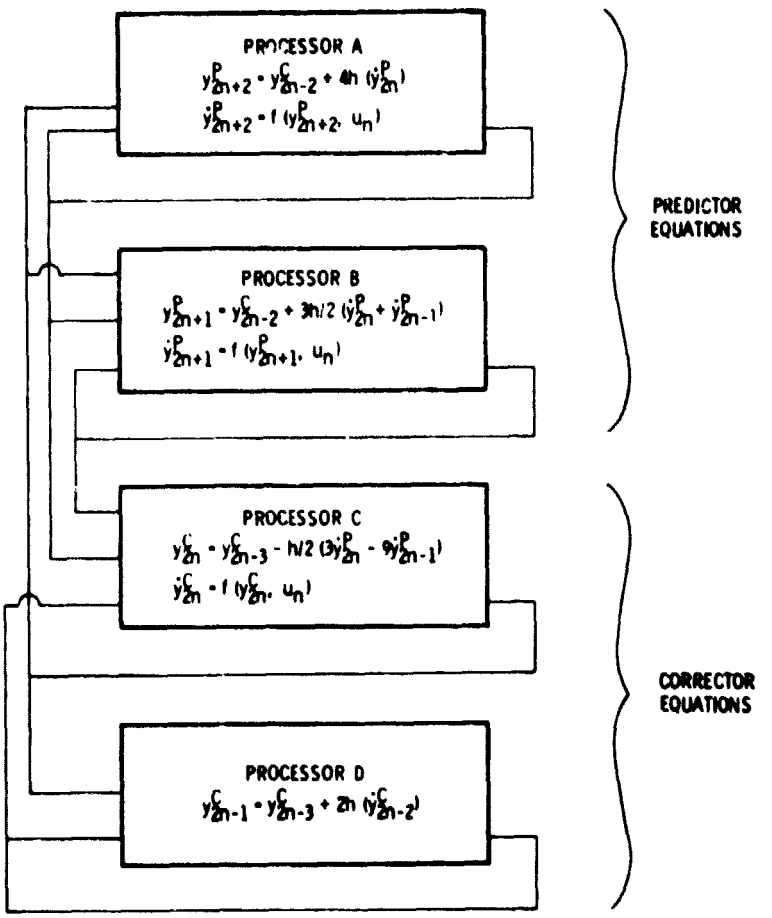

Figure 1. - Implementation of four processor parallel predictor-corrector integration algorithm.

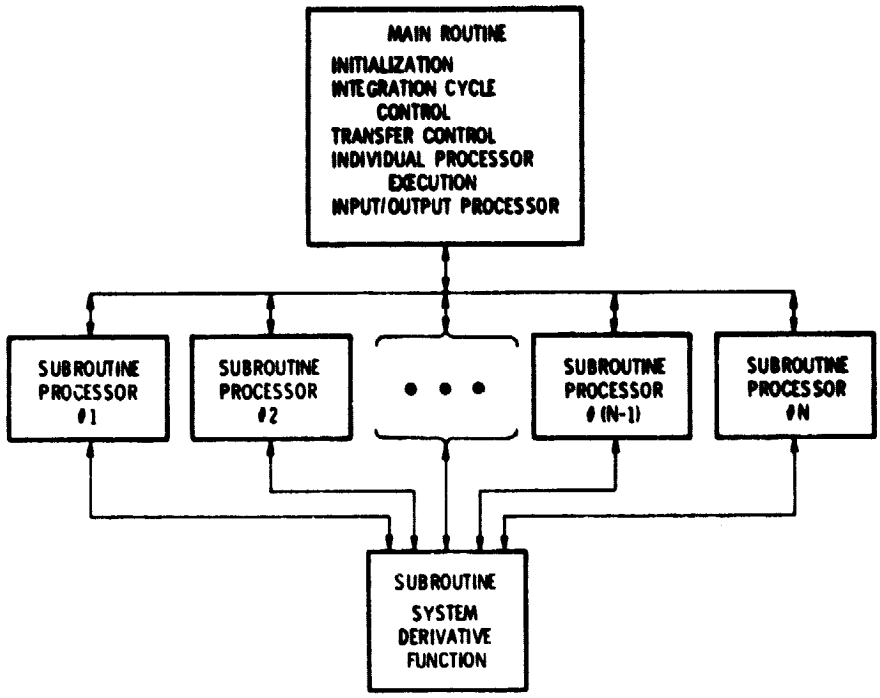

Figure 2. - Structure of software simulator.

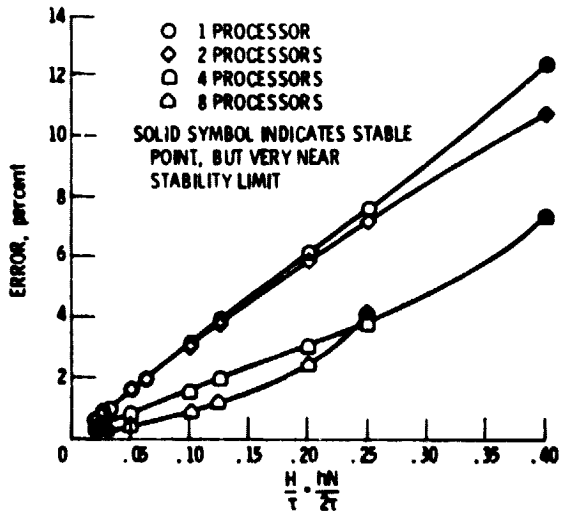

Figure 3. - Relative error for first order system: 1/(τs + 1).

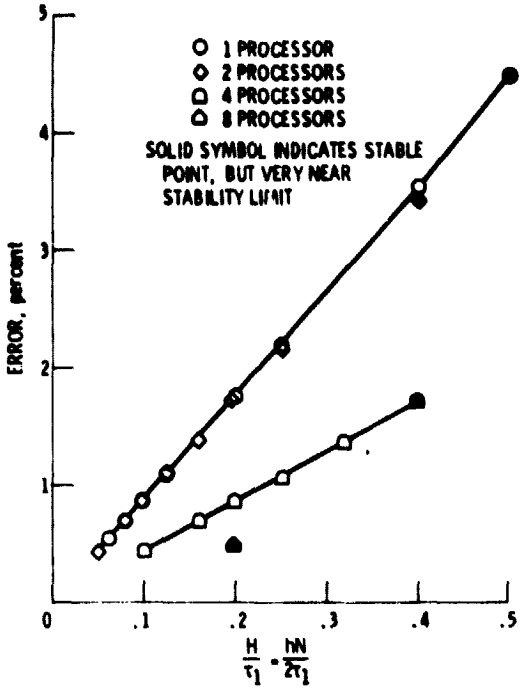

Figure 4. - Relative error for second order system: $1/(\tau_1 S + 1)(\tau_2 S + 1)$; $\tau_1 = 0.01$ sec, $\tau_2 = 0.02$ sec.

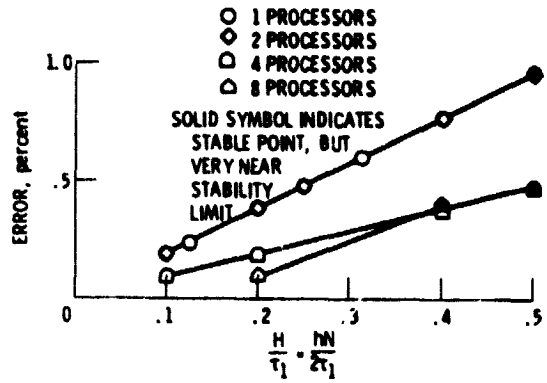

Figure 5. - Relative error for second order system: $1/(\tau_1 S + 1)(\tau_2 S + 1)$; $\tau_1 = 0.002$ sec, $\tau_2 = 0.02$ sec.

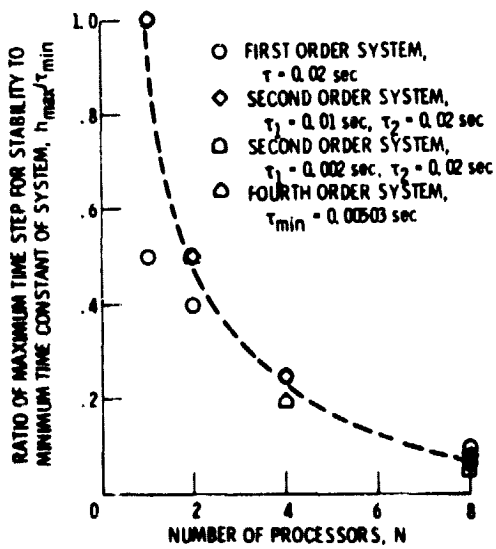

Figure 6. - Maximum stable time step with respect to minimum time constant of system for given number of processors.

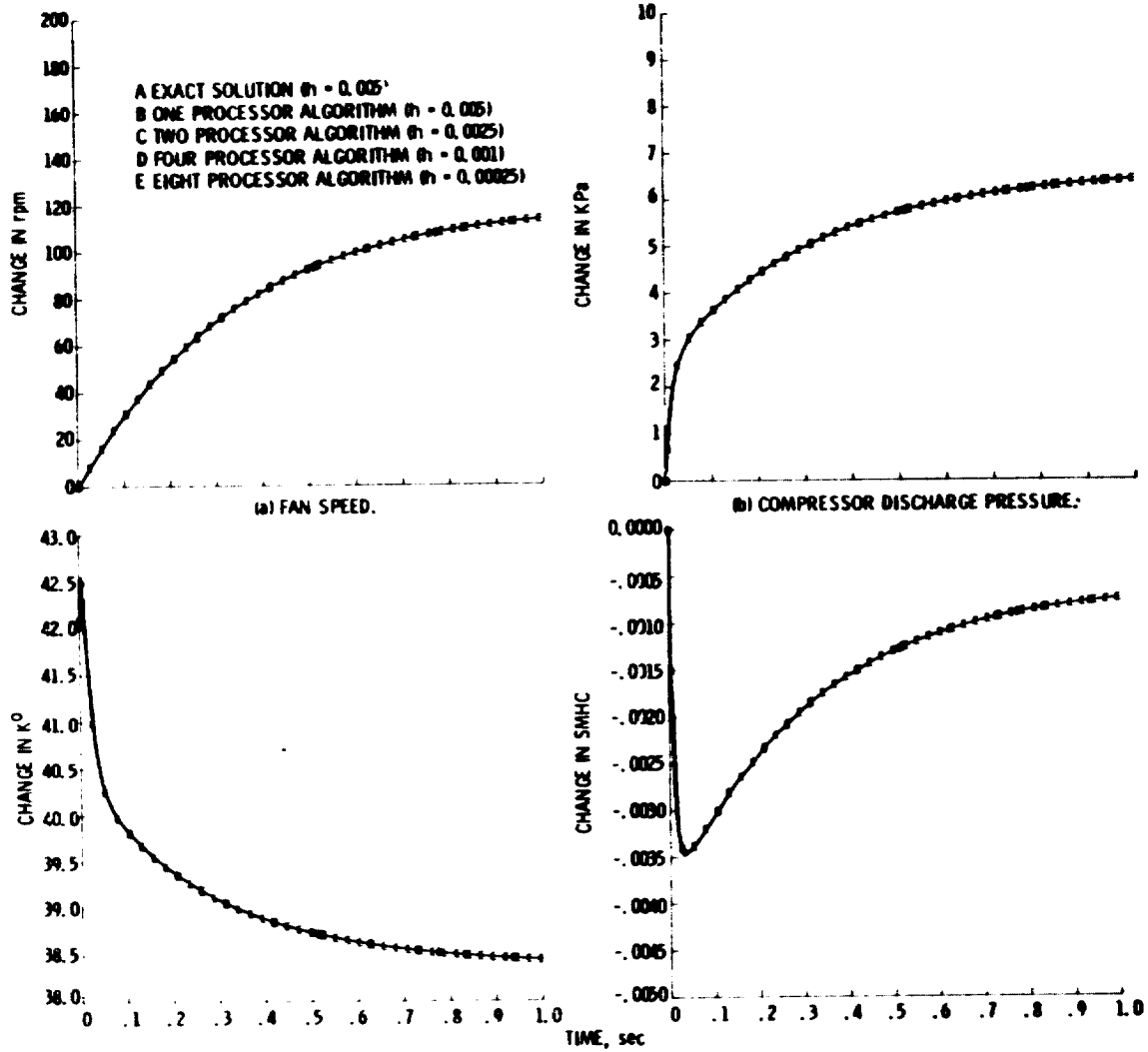

Figure 7. - Effect of number of processors on engine model response to a +3 percent step in fuel flow for 1.0 sec.

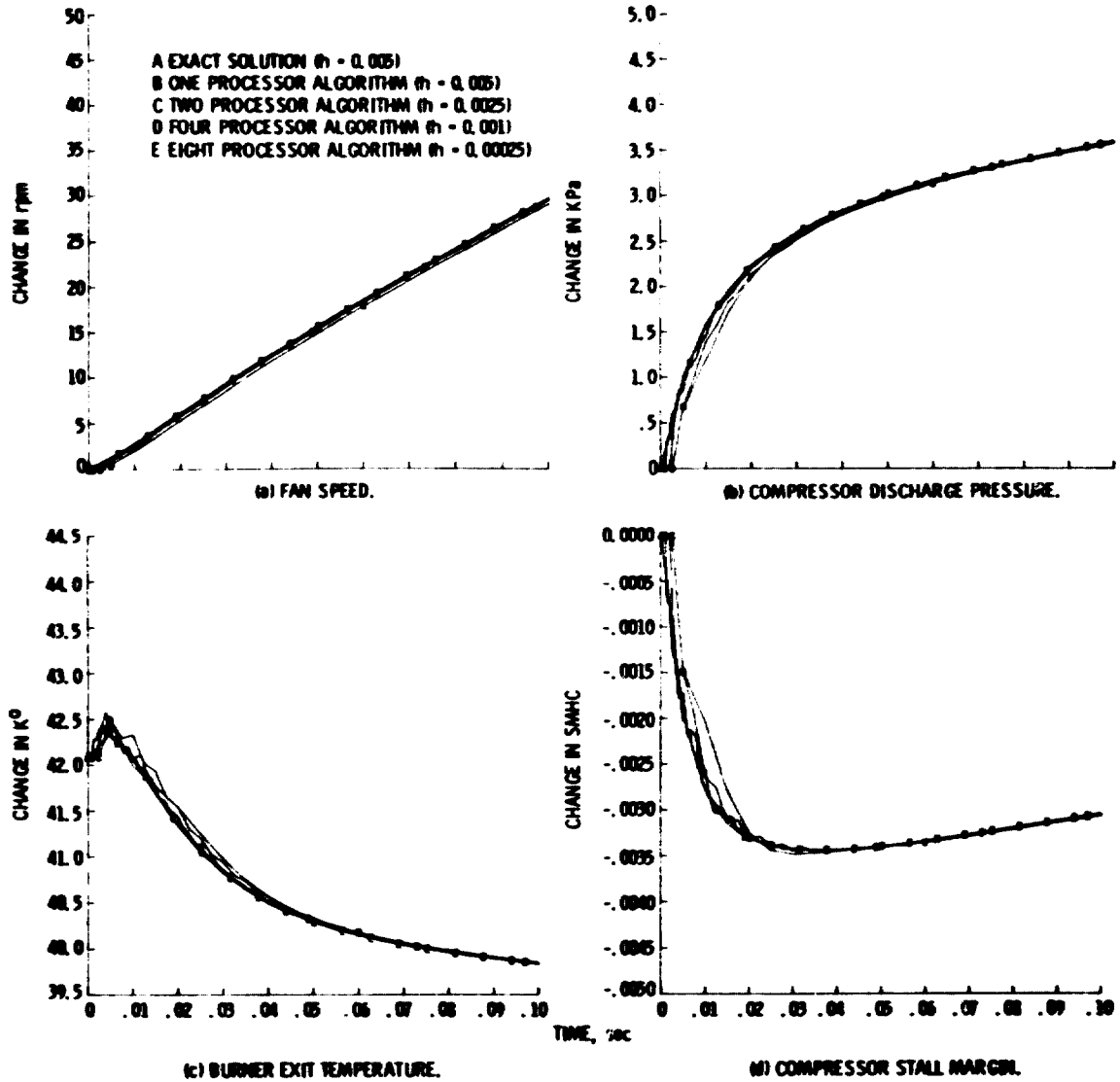

Figure 8. - Effect of number of processors on engine model response to a +3 percent step in fuel flow for initial 0.1 sec.

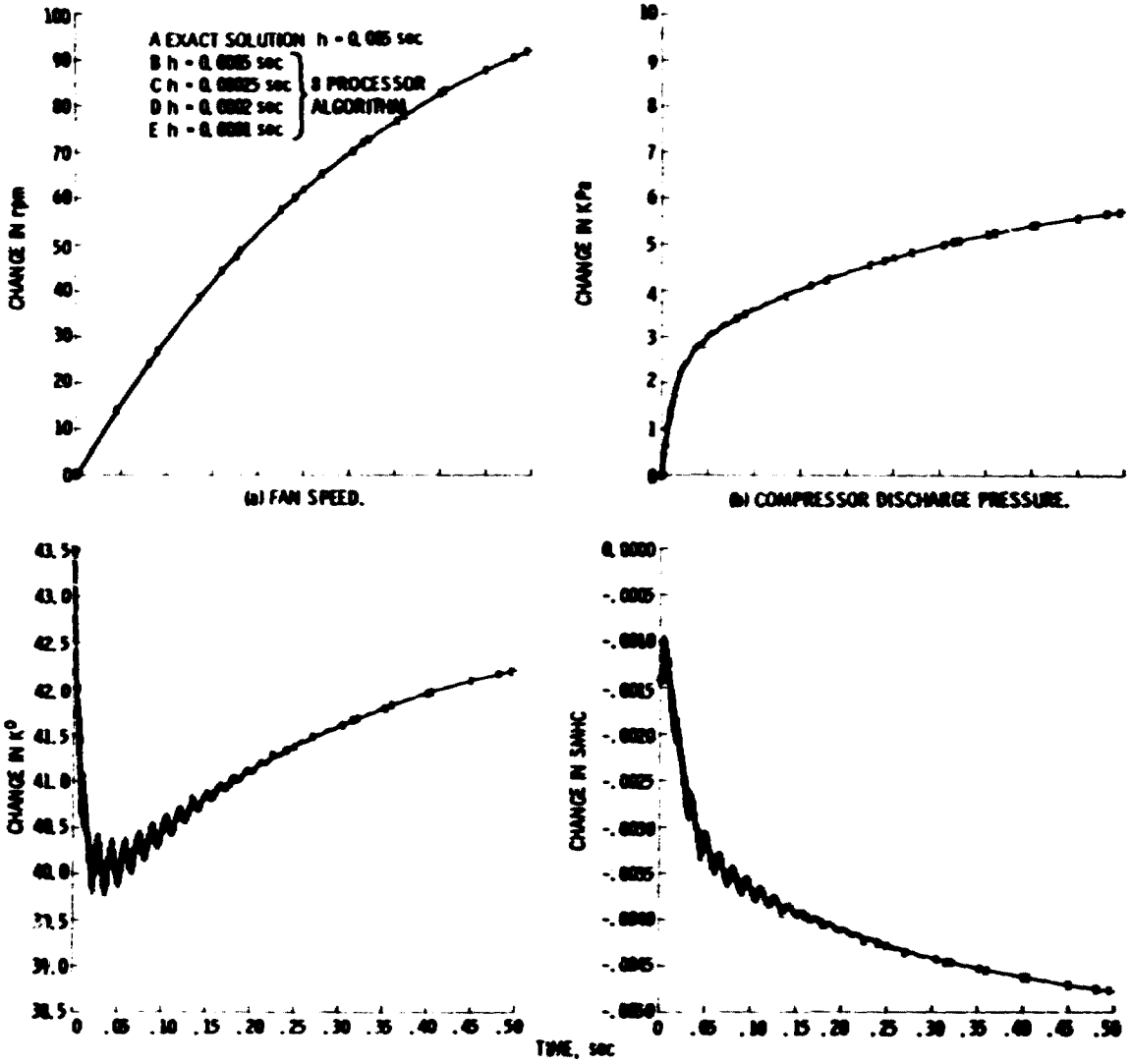

Figure 9. - Effect of step size on engine model to a +3 percent step in fuel-flow for 8 processor algorithm.

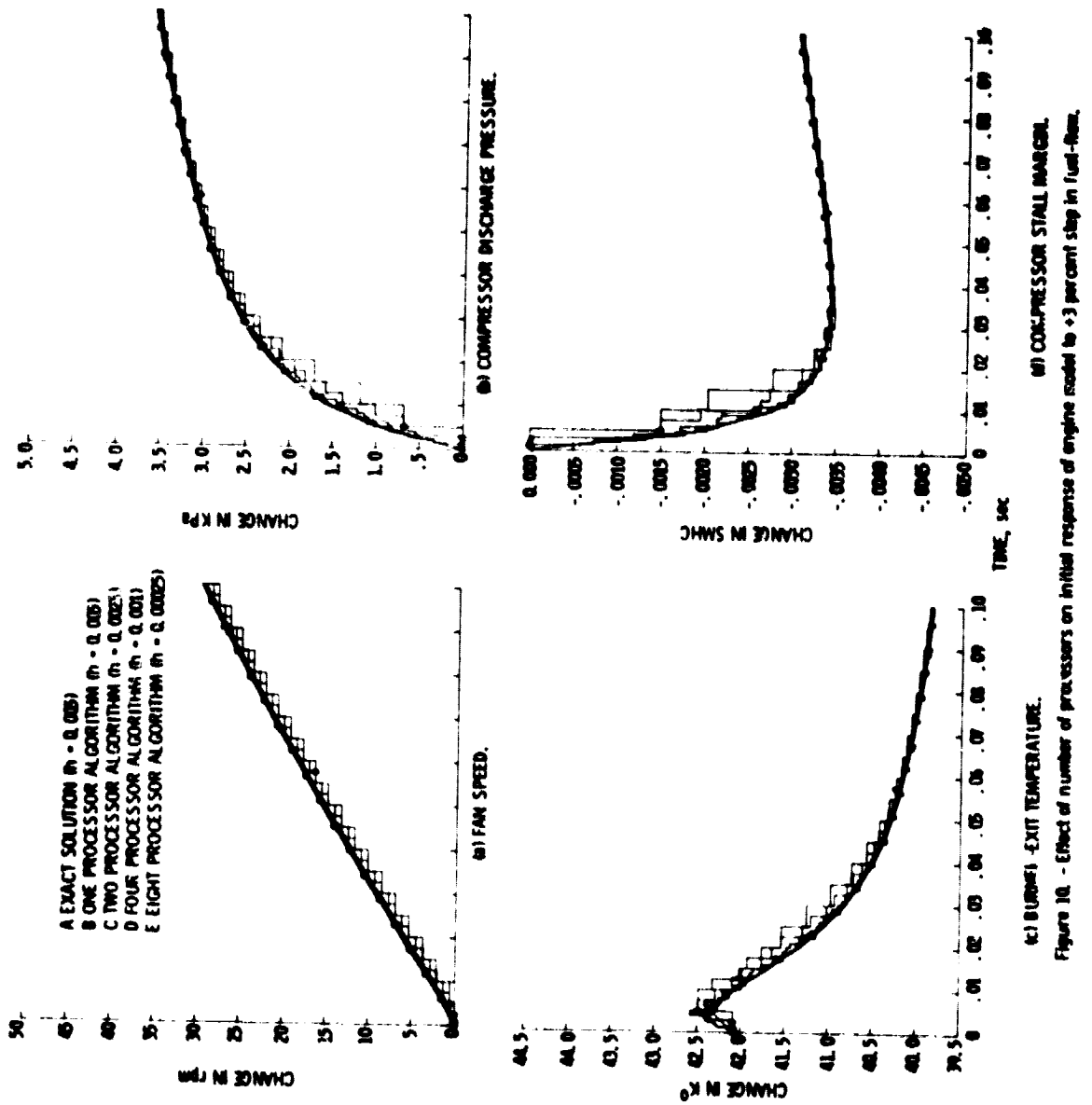

Figure 10. - Effect of number of processors on initial response of engine model to +3 percent step in fuel flow.